GOVERNED MEMORY INFRASTRUCTURE FOR REGULATED AI

Technical Security & Architecture Brief

Runtime authority verification, separate action admissibility, two-stage retrieval governance, tokenized memory, and cryptographically verifiable evidence — engineered as infrastructure, not a wrapper.

Field	Detail
Version	4.0 · August 2026 (supersedes 1.0 and its addenda)
Prepared for	Enterprise CISOs, CTOs, security architects, AI governance teams, technical investors, and acquisition due-diligence reviews
Basis	Statements describe the implementation present in the current source tree and database migrations at the time of writing. Roadmap items are labelled as such.
IP status	Patent Pending — U.S. Provisional Patent Application No. 64/137,172, filed August 20, 2026. Title: Governed Memory and Execution Control with Cryptographic Authority Lineage, Material-State Admissibility, and Disclosure-Time Effect Boundary.
Classification	Confidential — technical brief. Contains no source code, secrets, key material or environment configuration.

01 · Executive summary

Trace Continuity is governed memory infrastructure for regulated AI. It sits between an application or agent and the data it wants to remember, and enforces at execution time — not at policy-review time — who may write, who may read, what may be stored, how identifiers are protected, and how every decision is proven afterwards.

There is exactly one server-side path to governed data: the execution gate. Authority verification, action admissibility, governance, matter boundaries, tokenization and evidence are non-optional stages on that path.

Key properties

- **Single enforcement point.** One execution gate. A build-blocking scanner parses the whole source tree with the TypeScript compiler and fails the build if any file outside the gate reaches governed storage — under any quoting, template literal, computed table name, governed RPC call, or re-exported privileged client.
- **Authority is verified, not assumed.** Credentials are pinned to a single authority. Delegation, inheritance, genesis lineage and the per-tenant authority-event chain are all verified at the moment of use and all fail closed.
- **Admissibility is a separate decision.** Valid authority does not imply an admissible action. Material conditions are evaluated independently and audited on their own.
- **Retrieval is governed twice.** Preparation admissibility before anything protected is fetched; a freshness re-read immediately before decryption; a fresh final re-evaluation at the effect boundary, which can deny or tighten and discards the prepared payload.
- **Evidence is chained and sealed.** Per-action hash-chained audit entries, an HMAC-sealed action receipt, and a disclosure receipt carrying a keyed fingerprint of the exact payload approved to leave the boundary.
- **Tokenized memory.** Detected identifiers are encrypted into a per-tenant AES-GCM vault; the governed store holds opaque tokens. Detokenization is a distinct right and is audited.

What changed in version 4.0
Version 1.0 described a six-step authority check, a single retrieval decision, and an append-only audit table. All three descriptions are now out of date. The August 2026 hardening added credential pinning, deterministic multi-candidate evaluation, runtime authority-chain verification, atomic condition evidence, two-stage retrieval governance with a pre-vault freshness check, per-action hash chaining, sealed action receipts and disclosure receipts. Sections 3 through 9 below replace the earlier text.

02 · The problem this exists to solve

Memory frameworks solve remembering. Trace Continuity solves governance of remembering — the capability required to put AI memory into a hospital, a defense supplier or a law firm without violating HIPAA, CJIS, ITAR or attorney-client privilege.

Failure mode	Typical stack	Trace Continuity
Authorization	Decorators and policies at the app layer, easy to skip	One execution gate; an AST scanner fails the build on bypass
Auditability	Application logs, mutable, PII-leaking	Append-only, per-action hash-chained entries closed by a sealed receipt
Identifiers	Regex in one endpoint, absent in the next	Tokenization on every write, re-scan on every read
Revocation	Role change with no runtime effect	Revocation denies at next use; a pinned credential dies with its authority
Changing facts	Not modelled at all	Material conditions evaluated separately, with revisioned hash-chained history
Tenant isolation	RLS only	Gate + RLS + per-tenant vault key + lineage tenant checks

03 · Runtime authority verification

Authority verification runs inside the gate under the service role, against tables no authenticated client can read or write. Each step emits exactly one audit entry — pass, fail or skipped — so a denial always shows which step failed and why.

Step	What it establishes
tenant_ownership	The caller is a member of the tenant being acted against.
credential_binding	If the caller presented an API key, that key is pinned to the one authority it was issued against. That authority — and no other — is evaluated. Revoked, expired, deleted or foreign-tenant pinned authority denies the request. Session callers are recorded as unbound.
scope_validation	Some held authority carries the scope the action requires.
candidate_evaluation	Where several held authorities could apply, candidates are ordered deterministically (created_at, then id) and evaluated one at a time through the complete sequence. The first that passes end to end is selected; every rejected candidate and its reason is recorded. Skipped entirely when a credential is pinned.
delegation_chain	Every link in the delegation lineage is present, unrevoked, unexpired, within the same tenant, acyclic and within depth. Any other outcome is an explicit named failure, never a silent pass.
inheritance_resolution	The same discipline applied to inherited authority.
execution_rights	This authority holds the specific right being exercised.
genesis_link_verification	The selected authority's lineage terminates at the tenant's immutable genesis root, does not cross a tenant boundary, and the stored genesis hash still recomputes.
authority_chain_integrity	The tenant's authority-event hash chain is verified at request time at bounded cost (see §05). Divergence — or an unverifiable result — denies the action.
final_decision	Allow or deny, recorded with the granting authority and its genesis link.

Credential revocation actually terminates the credential
Before hardening, a revoked key-bound authority could be silently replaced by another authority held by the same actor. The caller context now carries the bound authority id, and a pinned credential is evaluated against that authority alone. The audit entry records the binding as pinned or unbound.

04 · Action admissibility — the second, separate decision

Authority answers whether the actor may act. Admissibility answers whether, given the material conditions in force at this instant, the action is permissible even though the authority behind it remains perfectly valid. Both are evaluated inside the same gate, under one action id, and recorded as distinct evidence. Admissibility never substitutes for, short-circuits or relaxes authority verification.

Condition facts and their evidence

Material state is stored as condition facts: a scalar bound to a tenant, a subject (tenant, actor, matter or record), a condition key, a source, an observation time and an optional expiry. An assertion is a single atomic database operation. It takes a transaction-scoped advisory lock keyed on the subject and condition key, allocates the next revision inside that lock, reads the prior chain hash under the same lock, and writes both the current-state row and the immutable hash-chained history row in one transaction. A unique index on (tenant, subject, condition key, revision) aborts a duplicate revision rather than forking the chain. A verification function replays a chain and names the first divergence.

Consequence: there is no window in which a condition is in effect without its evidence row. Evidence-before-effect is a transactional property, not a convention in application code.

Fail-closed semantics

- A rule whose fact is missing denies by default. Allow and skip are explicit, per-rule opt-outs.
- A malformed, unparseable or unknown-operator rule denies. There is no expression interpreter and no user-supplied code.
- An expired fact is treated as unestablished.
- Caller-supplied per-request state is validated, restricted to scalars, labelled as untrusted in the record, and can only make an action inadmissible — never admissible. Rules marked trusted-only ignore it entirely.
- Any exception inside the evaluator becomes an audited denial.

Condition-to-action correlation

Every determination records the facts it relied upon: fact id, condition key, subject type and id, revision, value, source, observation time, rule origin, and the id of the action that asserted that exact revision. The asserting action is resolved from the condition history by fact id and revision and labelled correlated, pre-correlation-history (history that predates correlation, or an assertion with no recorded action), or unavailable (the lookup itself failed). Correlation is evidence enrichment only; a failure is recorded and can never make an inadmissible action admissible.

05 · Authority origin and runtime chain verification

Every tenant has one immutable genesis record establishing the origin of authority, and an append-only per-tenant event chain recording every authority change: creation, scope change, execution-rights change, delegation or inheritance change, expiry change, revocation, restoration. Events are written by a database trigger on the authorities table, so a change that skips the application layer still leaves an event. Each event carries the previous event's hash and its own, computed as SHA-256 over canonical bytes.

What hardening added

- A chain head row per tenant — head event, head hash, event count — maintained by trigger in the same transaction as each event insert, so a removed or injected event immediately puts the recorded head and count out of agreement with the table.
- An append-only ledger of verified checkpoints, itself immutable and non-truncatable.
- A full replay function that recomputes every event hash, asserts linkage, ordering and count continuity, and returns the first divergence.
- An incremental verifier used on the request path: it compares the stored head to the actual tip and verifies the most recent window of events plus checkpoint linkage — bounded work, independent of history size.
- A checkpoint is only written after an actual full verification passes. The first checkpoint is not trusted on assertion.

During baseline establishment, a small number of legacy tenants were found to carry a null previous hash on their first event, caused by a race between signup and genesis backfill. Rather than paper over it, the verifier accepts a null previous hash only for the root authority's creation event and labels that state explicitly. Every other null is a divergence.

Threats this addresses

- **Fabricated root authority.** Lineage must terminate at the genesis root, not at any root.
- **Cross-tenant re-parenting.** The lineage walk rejects the moment it crosses a tenant line.
- **Edited origin record.** The recomputed genesis hash no longer matches and the runtime denies until it is investigated.
- **Rewritten history.** Triggers block UPDATE, DELETE and TRUNCATE; the chain would break at the tampered row; and the request-path verifier detects head and count disagreement.

06 · Tokenization and identifier protection

Detection runs on every write. Detected values are encrypted with the tenant's AES-GCM vault key and stored in the vault; the governed record holds only opaque tokens.

- Vault keys are per-tenant. A token from one tenant cannot be decrypted with another's key.
- Detokenization is a distinct right the authority must hold, and every detokenization is audited.
- Retrieval re-scans stored payloads, so an improvement to the classifier becomes immediate containment of values written under an older one; anything newly recognised is redacted.
- Values the classifier cannot categorise are recorded with an explicit review reason rather than passed through silently.

07 · Two-stage retrieval governance

The read path has three distinct governance points rather than one.

Point	What it does
Stage A — preparation admissibility	Runs after the record is located and the retrieval-time authority re-check passes, and before any protected value is fetched or decrypted. Its only job is early refusal. A Stage A denial ends the request with a complete evidence trail and no vault access at all. Its audit detail is explicitly marked preparation-only so it can never be read as the disclosure decision.
Pre-vault freshness re-read	Immediately before decryption, authority, matter boundary and ethical wall, and material conditions are re-read from the database. If any has changed, the request is denied at this point: no ciphertext is fetched and no decryption occurs.
Stage B — final admissibility (effect boundary)	Runs after every operation that can still change the response and before serialization. It re-executes — not replays — authority verification including genesis and chain integrity, admissibility evaluation, and the matter, wall and hold check against the database at that instant.

Stage B can deny outright, or tighten a transformation. Transforms may only tighten between A and B; a transform newly required at B over an already-detokenized field blocks disclosure. On denial the prepared payload — including any plaintext already decrypted and held in memory — is discarded and never returned. The boundary record states the boundary timestamp, the disposition, the authority basis at that instant, conditions relied upon and conditions not incorporated, transforms, transforms added since preparation, whether the prepared payload was discarded, and the residual time-of-check window.

08 · Evidence: chained, sealed, and fingerprinted

Every operation produces an ordered set of audit entries tied by one action id. Three distinct guarantees sit on top of that record; each addresses a different adversary.

Guarantee	Mechanism	What it actually proves
Append-only	Triggers block UPDATE, DELETE and TRUNCATE on the audit log, condition history, authority events, genesis, checkpoints, action receipts and disclosure receipts.	Ordinary application code and ordinary database roles cannot alter or remove evidence.
Hash chaining	Each audit entry stores the prior entry hash and its own, written by a BEFORE INSERT trigger under an advisory lock keyed on the action. A verification function recomputes every entry and names the first divergence.	An edited, reordered or removed interior entry is detectable. On its own, chaining does not detect removal of the tail of a chain, and an adversary able to rewrite the whole table could recompute every hash.
Sealed receipts	The action's final chain head is sealed with HMAC-SHA256 under a purpose-separated key and stored as an immutable action receipt.	This is what makes tail truncation detectable: the receipt binds the final entry hash and cannot be recomputed without the key, so a shortened chain no longer matches its receipt.

Disclosure receipts

At the effect boundary, a disclosure receipt records the boundary facts and a keyed HMAC-SHA256 fingerprint of the canonicalised final payload. The payload itself is never stored. An auditor holding a candidate payload can confirm it is the exact representation the gate approved. Stated precisely: the receipt attests the representation approved at the boundary and that the binding facts were not altered afterwards without the key. It does not by itself attest what the network transport ultimately delivered to the caller.

Key custody

Receipt keys are derived by HKDF-SHA256 from a single deployment-held root secret, read from the server secret store at call time, under three distinct purpose labels: action receipt, disclosure receipt, and disclosure fingerprint. Purpose separation means a receipt from one domain can never be replayed as a receipt of another. The keys are held by the deployment; they are not per-tenant. A non-secret key identifier is recorded with each receipt so an auditor can tell which key generation produced it without the key being revealed. Custody is reported on the receipt as environment secret store or unavailable; when the key is unavailable the gate denies disclosure rather than release an unattested payload. A managed KMS or HSM custody mode is a declared future state, not a current one.

09 · Execution gate and bypass prevention

The gate is a single server-only module. Every server function and public API route that touches governed data calls it. Any exception at any step becomes an audited denial.

The bypass scanner

The build-blocking scanner parses every TypeScript and JavaScript file in the project with the TypeScript compiler and analyses the syntax tree. It reports direct access to the governed tables under any quoting style, table names built from template literals or computed expressions, calls into privileged governed database functions, re-exported service-role clients, and governed access occurring after the gate's final effect-boundary call. Only the gate module is allowlisted. The scanner runs automatically before every build and exits non-zero on any finding, so a bypass fails the build. It is also covered by its own tests, including a scan of the live source tree and a set of deliberately adversarial fixtures.

Database grants and row-level security remain the enforcement layer of record; the scanner is defence in depth against a future caller that reaches around the gate.

Connector fanout

Outbound connectors — SIEM, ticketing, notification — receive events after the gate decides. They cannot mutate a decision, cannot see raw identifiers and cannot re-enter the gate. Every fanout carries the operation label of the path that produced it, set explicitly at each call site: a read denial reports a read, not a write.

10 · Tenant isolation

Every table carries a tenant id. The gate binds each operation to the caller's resolved tenant before any database access; condition and authority queries are tenant-scoped and never relaxed; lineage walks reject a cross-tenant link; vault keys are per-tenant.

Cross-tenant authorization is no longer accepted by the production write gate. The cross-tenant attack scenario exercised by the public Break Arena runs through a separate, demo-only entry point invoked from the Break Arena module alone. The production gate strips the field at the type level and the public API handlers do not expose it. Tenant isolation has no production-reachable parameter that redirects authorization.

11 · Governance policy evaluation and drift

Policies are data. Lower priority number wins, deny beats allow at the same priority, and the default is deny. The same structured predicate engine backs both policy matching and admissibility rules — equality, ordering, set, text and temporal operators — so policies are not limited to strict equality. A malformed or unevaluable clause never matches.

Policy drift between the moment a record was written and the moment it is read is reported in three states, not two: **changed**, **unchanged**, and **no write-time evidence**. The absence of historical policy evidence is reported as unknown, never as drift.

12 · Legal governance

- **Matter boundaries.** Memory is bound to a matter. Tenant membership alone does not grant retrieval; an unknown matter is a denial, not an empty result.
- **Ethical walls.** A conflict screen is checked inside the gate, ahead of retrieval, and overrides access grants, roles and inherited authority. Raising and lifting a screen both require a reason and both stay in the record.
- **Legal hold.** A matter under hold is preservation-locked, and so is each individual record bound to it: a database trigger refuses the delete row by row, so a single governed record under a held matter cannot be removed. Records not bound to a matter are unaffected, and the hold state is recorded on allow and deny alike.

13 · Threat model — and its honest limits

Adversary	Outcome
Client or API caller attempting direct table access	No Data-API grants on governance tables to anonymous or authenticated roles; RLS as defence in depth.
Holder of a revoked credential	Pinned authority is re-checked at use; the denial cites the revocation.
Actor with a second, unrelated authority	A pinned credential is evaluated against its bound authority alone; no substitution.
Attacker forging lineage	Lineage must terminate at the genesis root within one tenant; hashes must recompute.
Attacker editing or deleting evidence rows	Blocked by trigger for ordinary roles; detected by chain verification; tail truncation detected by the sealed receipt.
Principal with full database ownership	Can disable triggers. Under this adversary the keyed receipts, not the tables, are what remain meaningful — provided the receipt key is not also held by that principal. This is stated, not defended against.
State that changes during a request	Re-read immediately before decryption and again immediately before disclosure; a change committed after that instant cannot affect an already-authorised disclosure.

Stated limits

Append-only protection is tamper-evidence, not absolute tamper-proofing. The residual time-of-check-to-time-of-use window is bounded to the serialization step; it is not zero. Trace evaluates the state that has been reported to it: material conditions enter through the conditions API, and no component polls an external system to discover a real-world change on its own. If a change is never reported, Trace cannot know about it. Security validation to date is internal; independent third-party assessment is a step we invite and expect a serious buyer to require.

14 · API and extensibility

- Every governed capability is reachable through the public API: governed write, governed read, condition assertion, condition history, condition reporting, and audit listing.
- Authentication is modular. Identity providers are registered through a common interface, so an enterprise IdP is an added module rather than a change to the core.
- Connectors are registered the same way and observe the gate rather than participating in it.
- Admissibility rules and policies are rows, not deployments. A new material condition is data.

15 · Verification and proof surfaces

The automated suite runs on every code change and the build fails on a gate bypass. Coverage includes the execution gate, tokenization and adjacency regressions, authority genesis, the hardening behaviours (credential pinning, cycles, cross-tenant links, depth overruns), matter boundaries, policy drift, connector labels, receipts and chain verification, the bypass scanner against adversarial fixtures, and a database-owner validation suite that exercises the protections directly at the database level.

Two proof surfaces are open to any evaluator at tracecontinuity.com behind a name and email gate: the Playground, which shows why a value was allowed, denied, tokenized or ignored; and the Break Arena, which invites direct attempts to defeat the governance layer.

Patent Pending — U.S. Provisional Patent Application No. 64/137,172, filed August 20, 2026. Trace Continuity Labs · tracecontinuity.com